Контролно 2
28 май 2020

Въпрос 1. Нарисувайте на лист хартия или в удобен за вас редактор на диаграми декомпозиция на модулите за описаната по-долу система. Качете диаграмата като стандартен графичен файл (например JPEG или PNG). Опишете така проектираната декомпозиция, като обосновете как точно избраната от вас архитектура ще удовлетворява изискванията.

Направете декомпозиция на модулите от архитектурата на софтуерна система за организиране на онлайн залагания на спортни събития, според дадените по-долу изисквания.

- Клиенти на системата са различни организатори на залагания, които представляват юридически лица.
- Залагания може да се правят само от регистрирани потребители.
- Нерегистрирани потребители могат да преглеждат списъкът с налични събития за залагане и съответните предложения за коефициенти на залозите към тях.
- Коефициентите на залозите могат да се променят за секунди, като се изчисляват по математическа зависимост, която се определя количествено както от честотата на направените залози до момента, така и от прогнозни резултати.
- Прогнозните резултати се взимат от външна услуга, която се достъпва по специфичен протокол.
- Залаганията и печалбите не са в реални пари, в т.нар. кредити, които всеки потребител разполага в акаунта си. Кредитите може да се зареждат по различни начини, които се определят индивидуално за всеки клиент.
- Потребителите могат да правят неограничен бтой залагания, но само в рамките на текущо наличните им кредити.
- Системата е непрекъснато свързана с онлайн регистър на данъчната служда в съответната държава, където е регистриран клиентът.
- Системата трябва да е достъпна 24/7, тъй като потребителите могат да се включват от различни часови зони.
- Системата трябва да е защитена от външна намеса.

Отговор:

Описаната по-горе система и изисквания показват, че тя ще се използва както от юридически лица, като например bet365, eurosport и т.н., така и от физически такива. За да може да използват главните функционалности на този софтуер е необходимо те да бъдат регистрирани. Т.е. това очертава рамките на първия модул "Registration". За да може да се направи тази регистрация, възниква въпроса дали даденото юридическо или физическо лице изобщо съществува. Това би могло да се валидира чрез външна система на НАП или ЕСГРАОН съответно. Регистрацията ще отключва иначе забранени функционалности на системата, докато други ще са достъпни дори и без регистрация. Очевидно видовете регистрация ще са минимум два. Една от основните функционалности ще е преглеждането на опциите за залагания, която ще е достъпна и без регистрация и влизане в профила. Тези опции ще ги опаковаме в модула "Gambling List" (допълнително може да се върнем в този модул и да го декомпозираме рекурсивно надолу до разграничаване на секции). За изчисляването на коефициентите ще се грижи друг модул, който ще наречем "Gambling Algorithms". Той от своя страна според бизнес логиката ще се грижи да определя какви да бъдат коефициентите за залагане, така че юридическите ни потребители на системата да не са на загуба в никакъв сценарии, иначе ще се лишим от тяхното желание да използват системата ни. "Gambling List" ще си комуникира с още един модул "Bet", който е опаковал една друга основна функционалност на системата, валидна само за регистрирани физически лица, а именно самото залагане. Сега, възниква следния казус - кое ще се изпълни първо - актуализацията на коефициента за залога или приемането на залога от потребителя? За целта ще обособим друг модул "Gambling Scheduler", който ще играе роля на разпределител или някакъв шедулър, тъй като за тези процеси явно ще има борба за ресурси. Но нека се абстрахираме от това и допуснем, че операционната система ще се грижи за този проблем. Но тогава пък възниква друг въпрос - отлагане на свързването,

т.е. за всяка операционна система ще трябва да има обособен crack, patch или някакъв друг допълнителен софтуер, който да приспособява нашата система (явно или не за потребителя) към съответната операционна система. Това ще добави твърде много модули и ще лиши системата ни от контрол до някаква степен, за това нека се върнем на първоначалната идея с допълнителния контролер/разпределител. Тъй като прогнозите ще се взимат от някаква външна услуга, то нашата система ще си комуникира с API-то на тази външна услуга и то ще си комуникира с "Gambling Algorithms", която комуникация ще е предоставена със специфичен протокол. Сега, за да отговорим на следващото изискване - ще направим модул "Gamblers", в който ще има подмодули "User Info", "Account" и "Bets History". В него ще се зарежда информация за наличната сума на потребителя, както и на какво най-много обича да залага, колко и какъв тип залози прави, т.е. нещо като профил и история за него, но там ще се допитваме и когато той желае да направи някакъв залог но няма необходимите средства.

Тъй като системата трябва да е достъпна 24/7, то ще се наложи да реплицираме сървърите и различни сървъри да отговарят на различни клиенти. Например сървър за тенис залагания, сървър за футболни залагания и т.н. Може дори разделението да е по съвсем различен критетии - спрямо големината на залога, например. За да гарантираме, че няма да има загуба на данни при отказа, тоест да скрием отказа, ще поддържаме активен излишък като дуплицираме базата данни и я поддържаме активна при всяко обновление (ще използваме и журнали за улеснение). Това драстично ще намали производителността, но тук може и да се върнем да помислим още.

Нека разгледаме следващото изискване: За защита от външна намеса най-вероятно се има предвид DoS или DDoS атаките. Нека за улеснение разгледаме само първата.

- За целта ще въведем автентикация на потребителите. Например, ако само регистриран потребител може да използва дадени услуги на системата, то при регистрация ще трябва да разпознава в кое от квадратчетата на подадена снима има... част от жираф, например.
- Въвеждането на ограничение на експлоатацията на местата където са уязвими на атака. Например ако сървъра може да обработва по 1000 заявки за някакво изчисление, то всяко следващо ще чака. Но това пак не решава проблема с DoS SYN Flood типа досовска атака, където нападателят може да започва процедура по залагане но да оставя заявката без отговор и така те остават незавършени и заемат мрежови ресурс. За това може да се въведе и времеви лимит за изпълнението на всяка заявка, след което тя да се прекратява автоматично.

Някой по-сериозни мерски, които може да предложим и комбинираме:

- Наблюдаване на трафика с цел установяване възможно най-рано на DoS атаката. За целта трябва да знаем какво означава ниско, средно и високо натоварване на системата, като отчитаме сезонност, маркетингови кампании и т.н. Така ще минимизираме загубите.
- Ще ограничим достъпа с допълнителен слой защита чрез средствата за ограничаване на по-нататъшен достъп като DMZ или firewall. Може при необходимост дори да използваме двойни защитни стени (периметърна и вътрешна).
- Ако допуснем, че приложението генерира сериозен интерес и трафик от различни части на света, то ще съхраняваме данните на няколко сървъра по целия свят. Това го правят именно CDN мрежите. Те услужват данните към потребителите от сървър, който е най-близо до тях с цел по-бърза връзка, но това автоматично прави системата по-малко уязвима на такива атаки, защото има много други сървъри, които функционират.

Диаграма може да се направи по произволен начин, спрямо по-горе описаните модули и свръзки. Вариантите са десетки.

Въпрос 2. От какво зависи броят структури, които се включват в документацията на софтуерната архитектура. Защо мислите така?

Отговор:

Документацията включва няколко аспекта: как е организирана; каква е цялостната архитектура и защо е проектирана така. Тя трябва да може да се чете лесно, без никакво допълнително напрягане от гледна точка на специфика и сложност. Има за цел да разясни, а не да буди още въпроси и да има нужда от документация за документацията и така

рекурсивно да се заравяме в документации. Ако тя бива прекалено сложна има опасност да създаде повече проблеми, отколкото да разреши.

Документацията е предназначена за заинтересованите лица. Според това кои ще бъдат те, се определя какви структури трябва да се включват в нея. Според това коя структура ще представи архитектурата на-добре на съответното заинтересовано лице, тя трябва да бъде с приоритет при представянето на документацията.

Въпрос 3. Какво представляват т.нар. криви на полезността от CBAM и за какво се използват?

Отговор:

Кривите на CBAM са начин да се оцени при каква количествена характеристика на дадено качествено изискване, колко процента от потребителите на системата ше бъдат доволни. Типичен пример за това е кривата, която показва съотношението между процента хора, доволни от системата и времето ѝ за отговор. Разбира се вместо време за отговор може да се представи която и да е характеристика.

Тези криви се използват и са най-полезни, в случаите, когато имаме работеща система и се анализира архитектурата с цел намиране на възможни нейни подобрения. Знаем текущото състояние на системата и също така какво очакват потребителите от нея. Точното построяване на кривите на полезност е трудна задача, но в повечето случаи е достатъчно да се определят само 4 точки:

- текущо състояние на системата - времето за отговор на системата в момента и процента потребители, които са доволни;
- желано състояние - времето, за отговор, към което искаме да се впишем и процента потребители, които ще бъдат удовлетворени при него;
- най-добро - при какво време за отговор всички потребители ще бъдат доволни - това е т.нар. точка на насищане, подобряването на която не носи реално полза за системата;
- най-лошо - прагът за време на отговор, при който няма потребители, които да са удовлетворени от системата.

Разбира се, за по-лесно онагледяване използвахме пример за време за отговор, но това може да бъде всяка друга характеристика. При CBAM се създават такива криви за всеки един от главните сценарии за качество.

Въпорос 4. Какъв ефект може да има тактиката за активен излишък върху сигурността на ситемата?

Тактиката за активен излишък е начин за повишаване на наличността на системата. При нея се осигурява допълнителен ресурс - място за съхранение, изчислителна мощ и т.н. Идеята е, че вместо 1 сървър (който ни е напълно достатъчен, за да се справим със заявките които получаваме и трябва да обработим), да осигурим по-голям брой - например 3. Целта на тази тактика е тези сървъри да работят съвместно и да обработват едни и същи заявки. Ако настъпи срив при някой от сървърите, единственото, което трябва да се направи е да се отбележи кой сървър е активен. Тоест всички сървъри работят паралелно, просто се променя резултата от кой сървър ще се използва реално от системата. Ето защо тази тактика осигурява много добра наличност - дори 100% като разбира се има и своите минуси сред които освен високата цена и по-лошата сигурност. Тази тактика осигурява откриване на повече места, от които може да бъде атакувана системата.